Implementing a Prototype Metaqueue Using the Portable Batch System

James K. Cliburn, Jr.[†]
C. Stephen Jones[†]
Winfried H. Bernhard[‡]
William L. Asbury[‡]

Abstract

The U.S. Army Engineer Research and Development (ERDC) Major Shared Resource Center (MSRC) and the U.S. Air Force Aeronautical Systems Center (ASC) MSRC, with technical assistance and guidance from the National Aeronautics and Space Administration (NASA) Numerical Aerodynamic Simulation (NAS) Systems Division, have together successfully implemented a prototype metaqueue. The metaqueue is hosted on node subsets of three IBM SP computers (two at ERDC, one at ASC) and utilizes the NAS-developed Portable Batch System (PBS) as the batch mechanism for implementation. The purpose of the metaqueue is to demonstrate the feasibility of using PBS to move batch jobs from one MSRC to another when the computer at the job submittal site is too busy or is otherwise unable to run the jobs. The benefits of this activity include load balancing of participating systems, improved throughput of user jobs, pooling of resources for critical applications, and resource availability for users during outages. This paper presents the configuration of the ASC-ERDC MSRC metaqueue, a description of the components used, experiences with using the metaqueue, lessons learned, and challenges to future metaqueue efforts among Department of Defense high performance computing sites.

1 Introduction

Prime goals of a Major Shared Resource Center (MSRC) include maximizing resource utilization and minimizing user job queue wait time. Not uncommon are instances of load imbalance where one machine experiences a large number of queued jobs while another machine of the same architecture at the same or different MSRC remains underutilized. A method of transparently moving queued jobs from a busy system to an available system and back is highly desirable. For the purposes of this paper, such a method embodied within a batch system is called a *metaqueue*.

Another motivation for implementing a metaqueue arises whenever a target system is unavailable at one site because of hardware or other failure, lengthy ongoing upgrade,

[†] U.S. Army Engineer Research and Development Center (ERDC) Major Shared Resource Center (MSRC)
[‡] U.S. Air Force Aeronautical Systems Center (ASC) Major Shared Resource Center (MSRC)

network outage, or any of a host of reasons resulting in the non-availability of a desired system. A job submitted to a metaqueue would be transferred to an operational system at a participating site where the job would then execute, perhaps sending its results back to the originating site.

During periods of heightened military operational activity requiring dedicated MSRC resources, a metaqueue might prove useful in distributing high priority jobs among participating sites. This activity would effectively increase the pool of computational resources available to the DoD scientist, resulting ultimately in accelerated delivery of model and simulation products to the warfighter.

There are undoubtedly other motivations for implementing an operational metaqueue, and this paper reports the findings and results of a prototype metaqueue implemented at the ERDC MSRC and the ASC MSRC using the Portable Batch System (PBS). PBS is a batch system developed at the National Aeronautics and Space Administration (NASA) Numerical Aerodynamic Simulation (NAS) Systems Division. PBS is now maintained by Veridian, Inc., and invaluable guidance and technical assistance were provided by NAS and Veridian during the development of the ASC-ERDC prototype metaqueue.

2 Configuration

The ASC-ERDC metaqueue makes use of PBS executing on three IBM SP systems, one at ASC and two at ERDC. A small subset of the total node count of each system was configured out of the main compute pool and dedicated to the metaqueue effort. A separate instance of PBS was established in these dedicated partitions, while the remainder of the systems continued production work under another instance of PBS. SPs running PBS are particularly well suited to this isolation technique, since PBS allows easy exclusion of one or more nodes from the list of nodes to which jobs are assigned.

The configuration of each SP in the prototype metaqueue is presented in the following table.

System	MSRC	Total Nodes	Nodes in metaqueue	AIX version	PSSP version	PBS version
Hpc02	ASC	256	4	4.3.2	2.4	2.1p20
Osprey	ERDC	255	5	4.3.2	2.4	2.1p20
Pandion	ERDC	127	2	4.3.2	2.4	2.1p20

Portable Batch System

PBS is comprised primarily of three daemons (pbs_server, pbs_sched, and pbs_mom), a command set, and various support elements. In a typical installation a single server and scheduler execute along with one or more "machine oriented miniservers," or MOMs. In gross terms, the server establishes and maintains queues and access control lists, performs security checks, services requests arising from commands issued by users, and passes jobs to the MOM for execution. The MOM shepherds a job throughout its execution and monitors system resources. The scheduler implements site scheduling policies and requests that jobs be run at such time when sufficient resources are available and site scheduling policy constraints are met. Since scheduling policies can vary widely from site to site (or even from machine to machine within a site), users of PBS are encouraged to develop their own scheduler, and the server is equipped with hooks to enable a high degree of scheduler customization.

Both the ASC MSRC and the ERDC MSRC employ PBS on multiple HPC architectures as a production batch queueing system. Additional information pertaining to PBS is available at http://pbs.mrj.com.

Peer Scheduler

Arguably the most critical PBS component in the ASC-ERDC prototype metaqueue is the scheduler, called the *peer scheduler*. The peer scheduler (hereinafter referred to as simply the scheduler) was developed by NAS as part of a previous metaqueue effort within NASA, then modified to some degree as needed to operate within the MSRC domain. A set of configurable parameters is provided to the scheduler through a scheduler configuration file, which is read at either scheduler start-up or upon receipt of a particular Unix signal (SIGHUP). Many site scheduling policies and limits are defined in the configuration file, as are, in the case of the ASC-ERDC metaqueue, directives pertaining to the PBS servers (and schedulers) participating in the metaqueue. These participating systems are known as "peers," hence the name "peer scheduler."

An instance of the scheduler runs in each metaqueue partition on hpc02, osprey, and pandion. The scheduler operates in a "pull" fashion; that is, whenever a scheduler discovers that its system can accommodate additional jobs, it polls the PBS servers on its peer list in an effort to obtain additional work. If one of the peers has jobs queued, a job is moved (or "pulled") to the site that initiated the poll. The ASC-ERDC metaqueue peer directives for osprey are listed below.

```
PEER fr27n1e.wes.hpc.mil,:wes,primary,meta,LOCAL
PEER f1n5e.wes.hpc.mil,:wes,meta,primary,REMOTE
PEER f07n02s.asc.hpc.mil,:asc,meta,primary,REMOTE
```

The first directive identifies the "local" machine, in this case, osprey. The fr27n1e label is the host name of the PBS server node for osprey's metaqueue partition. Similarly, f1n5e and f07n02s are the host names for the corresponding metaqueue partition PBS servers on pandion and hpc02, respectively. In the directives above, osprey will poll the "primary" queue on both pandion and hpc02 and attempt to pull jobs into its own "meta" queue. The terms "primary" and "meta" are nothing more than queue names. The keyword at the end of each directive is self-explanatory: REMOTE implies a remote PBS server.

Of course, each participant in the peer group must have corresponding peer directive entries in its scheduler configuration file. To remove a participant from the peer group, simply comment out or remove the relevant peer directive.

In the ASC-ERDC metaqueue configuration, osprey could pull jobs from both pandion and hpc02, pandion could pull jobs from only osprey, and hpc02 could pull jobs from only osprey. A full three-way metaqueue could have been established with each machine able to pull jobs from all others, but this configuration would have added unnecessary complexity to the operation of the metaqueue.

3 Operation

The ASC-ERDC metaqueue was tested with a variety of applications, including CTH, a shock physics code developed at Sandia National Laboratory. Typical operation of the metaqueue consisted of submitting several jobs to a particular metaqueue partition, causing the nodes in that partition to become fully utilized and resulting in jobs queued but unable to run due to insufficient available resources on the local machine. A remote peer machine then detected the queued jobs and requested that the local server move the jobs, one at a time, to the requesting peer. This process continued until the remote peer was no longer able to accommodate additional jobs, at which time it ceased to poll the participant peers for jobs.

Site-specific Accounting Considerations

Each user job executed on an ERDC or ASC MSRC system accumulates node-hours (or CPU-hours) during execution. When a job completes, the accumulated execution time is deducted from the total number of hours allocated to the user's project by the appropriate allocating authority. ASC MSRC and ERDC MSRC differ in their implementations of this accounting process. At both sites there are often multiple users assigned to a single project, and less frequently, but not a rarity, a single user is a member of multiple projects. The alphanumeric string that uniquely identifies a project or sub-project, often called an ACID (account identifier), is of different length and format at ASC and ERDC. Additionally, a user's login name (user name) at ERDC might well be different at ASC. To overcome these differences, NAS developed a translator (called "acid mapper") whose sole function is to modify the project identifier and user name, if necessary, of a job on the sending host just prior to its move from one site to another. The acid mapper makes use of a flat file containing information about the user's accounts at both sites. Although not a general solution for the ACID and user name differences, it was suitable for the prototype metaqueue implementation.

4 Lessons Learned

While the implementation of the ASC-ERDC prototype metaqueue was a successful demonstration of transparent job movement from a user perspective, several items of interest emerged during its development and operation.

Version parity

To maximize the likelihood that a job submitted on one peer is successfully run with identical results on another peer using a third-party application code, the target codes should remain at the same revision level across participating sites. This notion also extends to operating systems and support software and, to a lesser extent, PBS itself. During the course of the development and implementation of the ASC-ERDC prototype metaqueue, all software elements used in the metaqueue operation were fixed at the same revision level.

Accounting information exchange
As previously mentioned, job accounting differences among participating sites exist and must be eliminated or accommodated as part of a production metaqueue. Currently, project allocations are dispensed for a project at a particular site on a particular architecture, and in a true metaqueue implementation a user might not possess an allocation at one of the participating sites. In that case, the node-hours accumulated at the job execution site must be transferred to the job origination site for proper posting to the user' s project. The ASC-ERDC metaqueue team designed and implemented a bare-essentials prototype relational database to exchange accounting information between the two sites.

To simplify the metaqueue process, ACIDs, user names, user IDs (UIDs), group names, and group IDs (GIDs) should be identical across participating sites. This presupposes a central clearinghouse for the assignment of these various identifiers.

Common user environment
Executables, libraries, compilers, applications, and the like are frequently located in different directories from site to site, increasing the probability that an arriving job will fail to find the necessary components required to run. A common scheme for directory names, default search paths, and file system structures would reduce this probability. For the purposes of the ASC-ERDC prototype metaqueue, all paths were well known and were hard-coded into job scripts.

Job tracking
To the ordinary user, a job that has been moved by the current version of PBS to other than its originating site appears simply to have vanished. Only if the user is aware of the potential sites where the job may run can he or she expect to find the job and track its status. Typically a user employs the *qstat* command to obtain job status. This is also the case with the ASC-ERDC metaqueue, except that if the job is no longer resident on the local PBS server, the user must specify the server on which the job is running in order to obtain its status. The PBS product is distributed with an X-based tool called *xpbs*, which enables a user to easily obtain job status among multiple PBS servers, but few users take advantage of it and its use is generally discouraged because of the additional load a number of concurrent xpbs windows places upon the network. A web-based queue management and monitoring tool would be highly desirable.

Data set movement
With currently implemented network technology it is impractical to move large files from one site to another. Jobs may require input files or produce output files in the hundreds of megabytes size range. In some cases, multi-gigabyte files are produced. Moving data files of this magnitude across the internet is a significant challenge.

Job failure investigation
A job can fail to execute or produce expected results for a wide variety of reasons. Under normal circumstances a user contacts the customer assistance center at the site where the job was submitted whenever problems occur. However, if the job starts and subsequently

fails at a site other than the submit site, difficulty arises in determining the troubleshooting authority for the job. Cooperation among participating customer assistance centers, shared call ticket systems, or web-based job monitoring tools could be employed to assist users in determining when, where, and why a job failed.

Kerberos

The current implementation of PBS is not Kerberos-aware, meaning that a job does not carry with it a Kerberos authentication ticket. In a production MSRC metaqueue environment, jobs would need the capability to carry an authentication credential.

5 Conclusion

The ASC MSRC and the ERDC MSRC, with guidance and technical assistance from NAS, have successfully implemented a prototype metaqueue system based upon PBS. The results of this work lead the team to conclude that an inter-MSRC metaqueue implemented in the fashion described in this paper, while feasible, is not practical at the current time. Of significantly more practicality is an intra-MSRC metaqueue, where multiple systems *within* a single MSRC move jobs about to relieve load imbalances and improve overall job throughput.